Test Automation Wokshop 2006 – Bond University

01 – Action Words

K. J. Ross & Associates Pty. Ltd.
PO Box 131, West Burleigh, 4219
Ph: 07 5522 5131 Fax: 07 5522 5232
training@kjross.com.au
http//www.kjross.com.au

Overview

- Automation Maturity
- Action Words
- Action Implementation
- SMARTLogger
- RDoc
- Exercise

Automation Maturity

1. Capture/Playback
2. Linked Scripts
3. Parameterised scripts
4. Data-Driven

- Maintenance overhead
- Lack of flexibility
- Difficult to integrate non-technical testers

5. Keyword-Driven / Action-Word

- Acknowledged best-practice

6. Model-Based

- Focus for research

Leverage Investment

Simple Capture Playback Demo

- Generated robot script

```
Sub Main
    Dim Result As Integer

    'Initially Recorded: 05/22/00  01:30:24
    'Test Procedure Name: Add 1 + 1

    Window SetContext, "Caption=Calculator", ""
    Window Click, "", "Coords=79,164"
    Window Click, "", "Coords=193,200"
    Window Click, "", "Coords=81,165"
    Window Click, "", "Coords=229,203"

    Result = LabelTC (CompareNumeric, "ObjectIndex=5", "CaseID=ADD1A;Value=2")

End Sub
```

Focus calculator window

click “1”

click “+”

click “1”

click “=”

Check value “2” shown in edit box

Web Testing

- Robot can be applied against the web browser as well
  - See demo handout

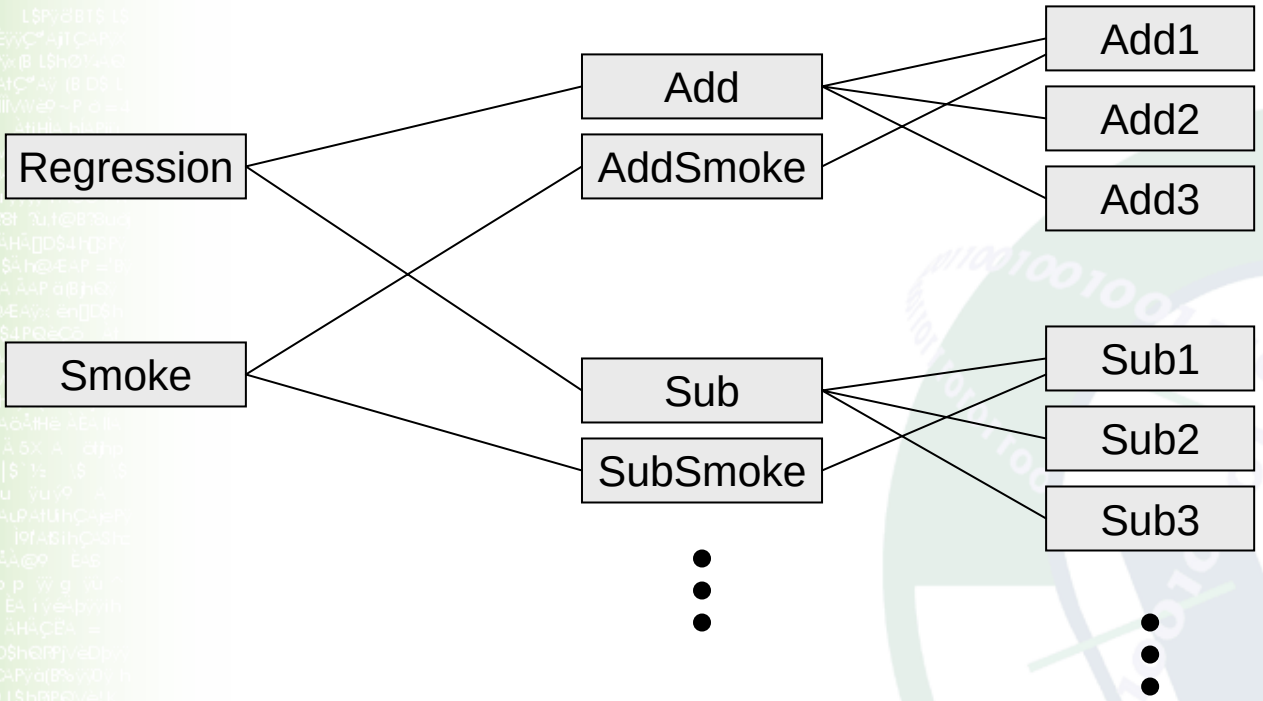
Resulting Script

Following script is generated:

```
Sub Main
    Dim Result As Integer

    'Initially Recorded: 13/04/2003 12:27:23 AM
    'Script Name: Temp
    StartBrowser "http://demo.smartlabs.com.au/smartcat/", "WindowTag=WEBBrowser"
    Window SetContext, "WindowTag=WEBBrowser", ""
    Window WMaximize, "", ""

    Browser NewPage,"HTMLTitle=SMART CAT - Welcome",""
    HTMLImage Click, "HTMLId=login", "Coords=79,13"

    Browser NewPage,"HTMLTitle=SMART CAT - Login",""
    EditBox Click, "Name=username", "Coords=32,10"
    InputKeys "admin"
    EditBox Click, "Name=password", "Coords=24,9"
    InputKeys "admin"
    PushButton Click, "Name=login"

    Browser NewPage,"HTMLTitle=SMART CAT - Menu",""
    Window ResetTestContext, "", ""
    Result = HTMLVP (CompareData, "HTMLId=Content", "VP=Object Data")
End Sub
```

- Opens a browser on defined page
- Maximise browser window
- Wait for browser to show the Welcome page
- Click on Login link
- Wait for browser to show the Login page
- Click on username form field
- Do a comparison of HTML object data

Linking Scripts



Parameterised Routines

- Build library routines that can be called by tests

```
Sub IntAdd(OpA,OpB,Res)
   Dim Result As Integer

   Window SetContext, "Caption=Calculator", ""
   Window Click, "", "Coords=236,59"
   InputKeys OpA & "{+}" & OpB & "="

    Result = EditBoxVP (CompareNumeric, "ObjectIndex=1", "VP=Alphanumeric;Value=" & Res)
End Sub


Sub Main
   Dim Result As Integer

   call IntAdd("1","1",”2")
End Sub
```

Data Driven Testing

- Script reruns test function for values specified in data file
- Some call them “datapools”

“1”, “1”, “2”
“1”, “0”, “1”
“0”, “1”, “1”

```
Sub dataDriven(filename As String)
    Dim operand1, operand2, result As String

    Open filename For Input As #1

    While Not EOF(1)
        Input #1, operand1, operand2, result
        SetupForTest()
        IntAdd(operand1, operand2, result)
        if fail then
                increment FailCount
        else
                increment PassCount
        end
        CleanupAfterTest()
      end
end
```

Action Word Testing

- Scripts are driven from test case data recorded in data files
  - Controls test execution based on commands
  - Accessible by non-technical testers
  - More effective for reuse / maintainability

Action Word Testing

- Scripts are driven from test case data recorded in data files
  - Controls test execution based on commands
  - Accessible by non-technical testers
  - More effective for reuse / maintainability

```
IntAdd, “1”, “1”, “2”
IntAdd, “1”, “0”, “1”
IntAdd, “0”, “1”, “1”

IntMult, “1”, “1”, “1”
IntMult, “1”, “0”, “0”
IntMult, “0”, “1”, “0”

IntDiv, “1”, “1”, “1”
IntDiv, “0”, “1”, “0”
IntDivError, “1”, “0”
```

Exercise: Data Driven Testing

- Write Robot VB code capable of driving calculator test data files. Assume:
  - no blank lines (e.g. the left-hand box)
  - the existence of Robot stub functions in the right-hand box.

```
IntAdd, 1, 1, 2
IntAdd, 1, 0, 1
IntSub, 0, 1, -1
IntMult, 1, 1, 1
IntMult, 1, 0, 0
IntMult, 0, 1, 0
IntDiv, 1, 1, 1
IntDiv, 0, 1, 0
IntDivError, 1, 0
```

```
Sub IntAdd(OpA,OpB,AddRes)

Sub IntSub(OpA,OpB,SubRes)

Sub IntMult(OpA,OpB,MultRes)

Sub IntDiv(OpA,OpB,DivRes)

Sub IntDivError(OpA,OpB)
```

Sample Solution: Data Driven Testing

```
Sub dataDriven(filename As String)
    Dim operation, operand1, _
      operand2, result As String

    Open filename For Input As #1

    While Not EOF(1)
      Input #1, operation, operand1,
    operand2
      If operation = "IntDivError" Then
        IntDivError operand1, operand2
      else
        Input #1, result
        If operation = "IntAdd" Then
          IntAdd operand1, operand2, result
        ElseIf operation = "IntSub" Then
          IntSub operand1, operand2, result
        ElseIf operation = "IntMult" Then
          IntMult operand1, operand2, result
        ElseIf operation = "IntDiv" Then
          IntDiv operand1, operand2, result
        Else
          MsgBox "Unknown operation in " & _
                 filename & ": " & operation
        Exit Sub
        End If
      End If
    Wend
    Close #1
End Sub
```

Overview

- Action Words
- Action Implementation
- SMARTLogger
- RDoc
- Exercise

Other Approaches To Action Words

- Data driven and Keyword Driven approaches offer more capability
  - Carl Nagle, “Test Automation Frameworks”
    - http://members.aol.com/sascnagl/Default.html
  - Linda Hayes, “Establishing and Automated Testing Framework”
  - Hans Buwalda, “Testing With Action Words”
    - http://www.cmg.nl
  - Many others

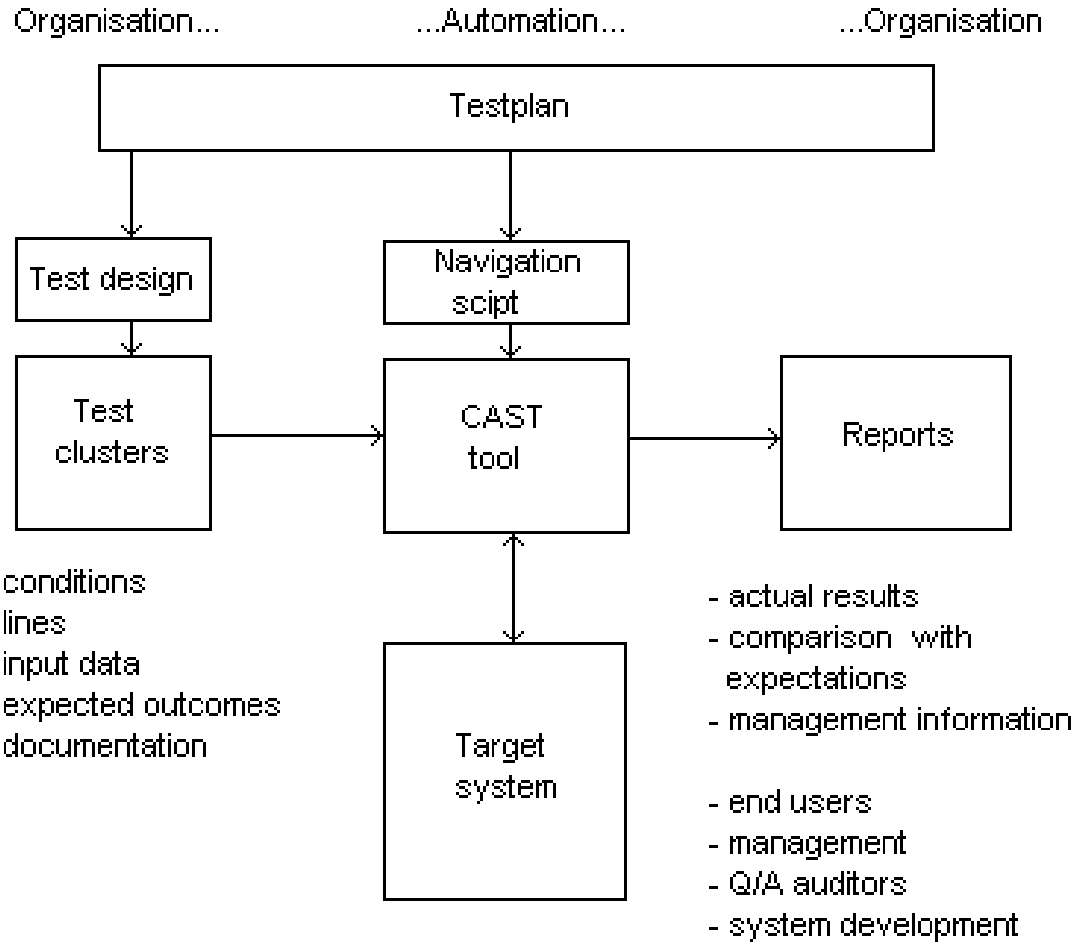
Test Automation Framework Guidelines

- Carl Nagle's guidelines:
  - Test automation is a fulltime effort, not a sideline
  - The test design and the test framework are totally separate entities
  - The test framework must be easy to expand, maintain and perpetuate
  - The test strategy/design vocabulary should be framework independent
  - The test strategy/design should remove most testers from the complexities of the test framework

Action Words

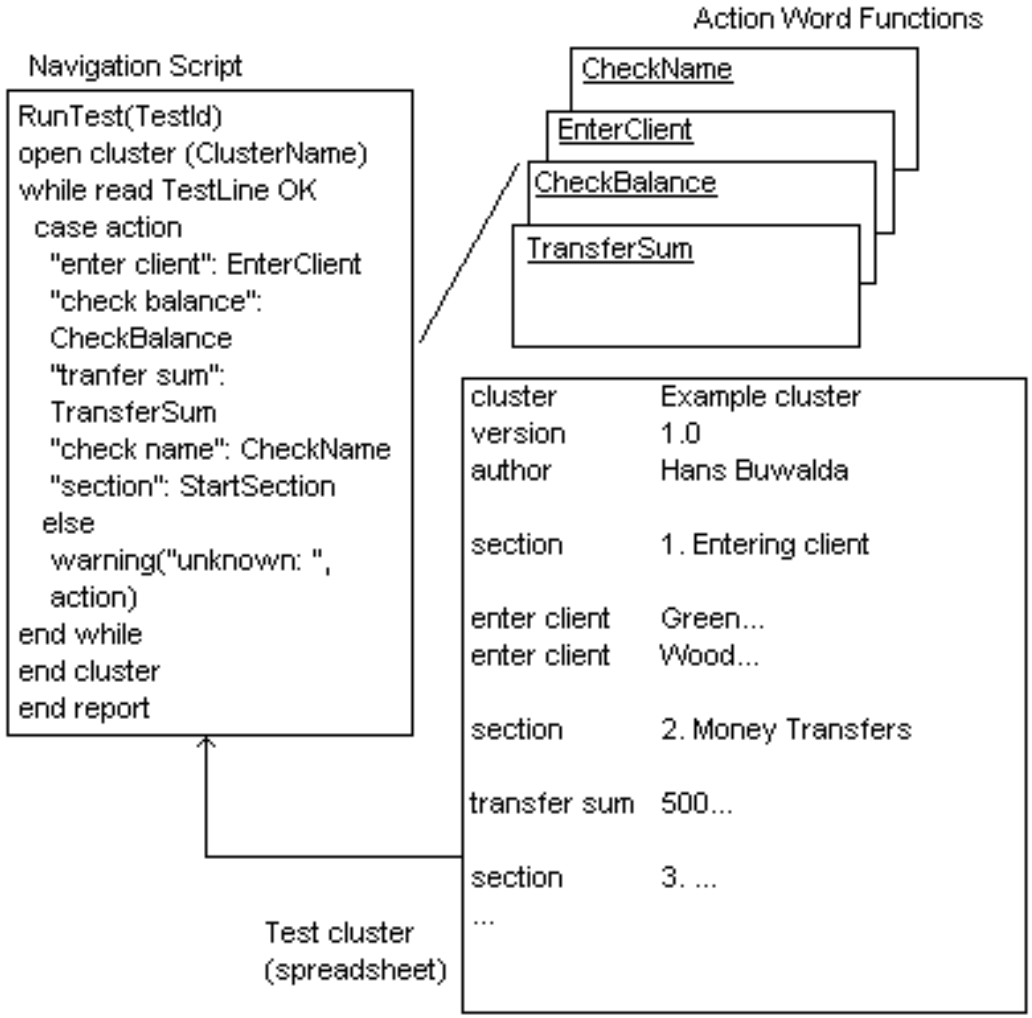
- Look at Hans Buwalda approach
- Representative of other approaches
- Separates the test cases from the test framework

Action Words Framework

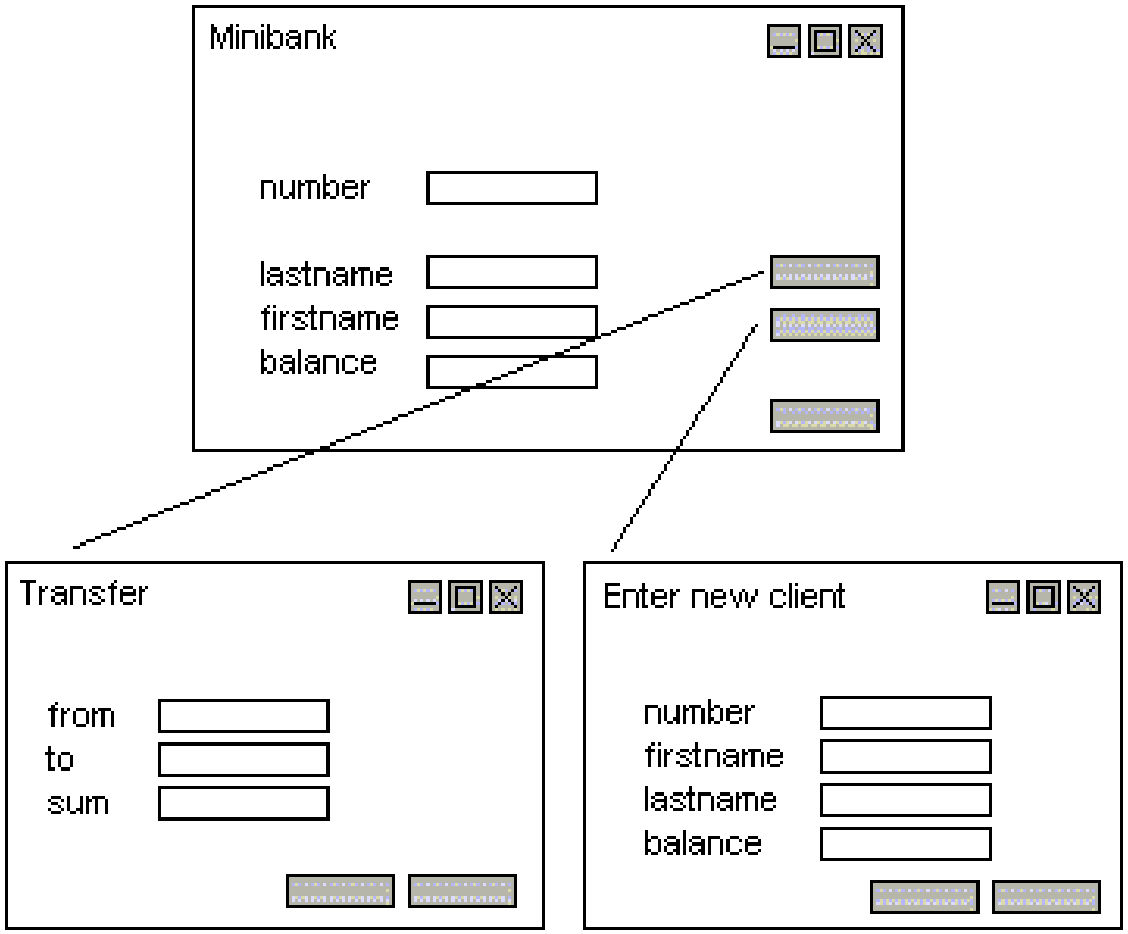


Action Word Elements

- Navigation script reads test cluster
- Test cluster calls action word functions
- Action word functions interact with application at lower level
- Test cluster uses spreadsheets
  - Benefits of spreadsheet functionality, e.g. formulas linking cell values



Example: Mini Bank



Case Study: Cluster

- Action words for
  - Enter client
  - Transfer
  - Check name
  - Check balance
- Note use of heading, sections and column names
- Focus on writing test cases
  - Could be written by non-technical tester
- Exported

	A	B	C	D	E
1	*cluster*	Example of a cluster			
2	*version*	1.0			
3	*author*	Hans Buwalda			
4					
5	*section*	1. Relation management			
6		lastname	firstname	number	balance
7	*enter client*	Johnson	John	500103381	1500
8	*enter client*	Juet	Marc	423137538	2100
9	*enter client*	Savy	Anne	848656467	1700
10	*enter client*	Puk	Piet	121003677	10
11					
12	*section*	2. Transfers			
13		from	to	sum	
14	*transfer*	500103381	423137538	500	
15	*transfer*	121003677	848656467	120	
16					
17	*section*	3. Checking names and balances			
18		number	lastname	firstname	
19	*check name*	500103381	Johnson	John	
20	*check name*	423137538	Juet	Marc	
21	*check name*	848656467	Savy	Anne	
22	*check name*	121003677	Puk	Piet	
23					
24		number	balance		
25	*check balance*	423137538	2600		
26	*check balance*	121003677	-110		

Case Study: Cluster

- Use of variables to keep and refer application generated data
  - Account number is required later

	A	B	C	D	E
1	*cluster*	Example of a cluster			
2	*version*	1.0			
3	*author*	Hans Buwalda			
4					
5	*section*	1. Relation management			
6		lastname	firstname	number	balance
7	*enter client*	Johnson	John	&keep[john]	1500
8	*enter client*	Juet	Marc	&keep[marc]	2100
9	*enter client*	Savy	Anne	&keep[anne]	1700
10	*enter client*	Puk	Piet	&keep[piet]	10
11					
12	*section*	2. Transfers			
13		from	to	sum	
14	*transfer*	&ref[john]	&ref[marc]	500	
15	*transfer*	&ref[piet]	&ref[anne]	120	
16					
17	*section*	3. Checking names and balances			
18		number	lastname	firstname	
19	*check name*	&ref[john]	Johnson	John	
20	*check name*	&ref[marc]	Juet	Marc	
21	*check name*	&ref[anne]	Savy	Anne	
22	*check name*	&ref[piet]	Puck	Piet	
23					
24		number	balance	* mistake in lastname	
25	*check balance*	&ref[marc]	2600		
26	*check balance*	&ref[piet]	-110		

Sample Output

- Pinpoint failures
- Provide summary

```
================================================
cluster name          : Example of a cluster
cluster version       : 1.0
cluster author        : Hans Buwalda

script name           : TestFrame Navigation Script
script version        : 1.0
script release date   : may 1997

run date and time     : 14-09-97 15:47:14
================================================

Section 1 - Relation management
1 (7):  enter client  Johnson   John    &keep john   1500

2 (8):  enter client  Juet      Marc    &keep marc   2100

3 (9):  enter client  Savy      Anne    &keep anne   1700

4 (10): enter client  Puk       Piet    &keep piet   10

Section 2 - Transfers
5 (14): transfer      89345591  15242252  500

6 (15): transfer      45673874  86318245  120

Section 3 - Checking names and balances
7 (19): check name    89345591  Johnson   John
                                Johnson   John
8 (20): check name    15242252  Juet      Marc
                                Juet      Marc
9 (21): check name    86318245  Savy      Anne
                                Savy      Anne
10 (22): check name   45673874  Puk       Piet
FAILED                          Puk       Piet
11 (25): check balance 15242252 2600
                                2600
12 (26): check balance 45673874 -110
                                -110

================================================
end of cluster        : Example of a cluster
finished at           : 14-09-97 15:47:19
time used             : 6

number of cluster lines : 23
number of test lines    : 12
number of checks        : 10
number passed           : 9
number failed           : 1
percentage passed       : 90%

failed at report lines:
10
================================================
```

Case Study: Action Word Functions

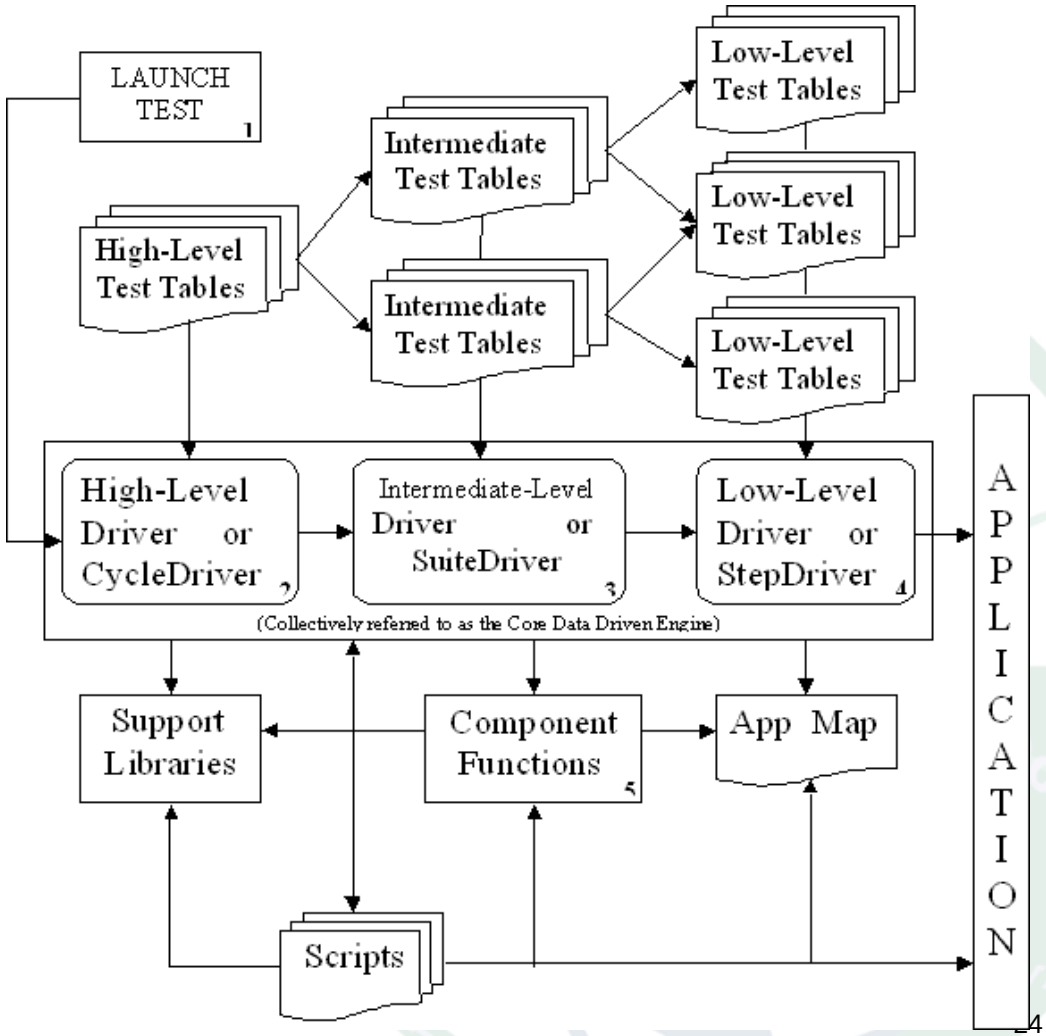
- Action word functions are implemented using automated test tool
- Library built up by tool specialist

```
EnterClient
PushButton "Relations"              * push the "Relations" button
WaitWindow "Enter New Client"       * wait for the entry window
EnterEdit "Number", arg[2]          * enter the values in the edit fields
EnterEdit "Firstname", arg[4]       * arg[2] etc are coming from the cluster
EnterEdit "Lastname", arg[3]
EnterEdit "Balance", arg[5]
PushButton "Process"                * will bring us back on the main screen
WaitWindow "Minibank Demonstration Program"
```

```
CheckBalance
* we don't have to go another screen
* so there is no PushButton statement here
EnterEdit "Number", arg[2]          * enter the account number
CheckEdit "Balance", arg[3]         * check the displayed balance
```

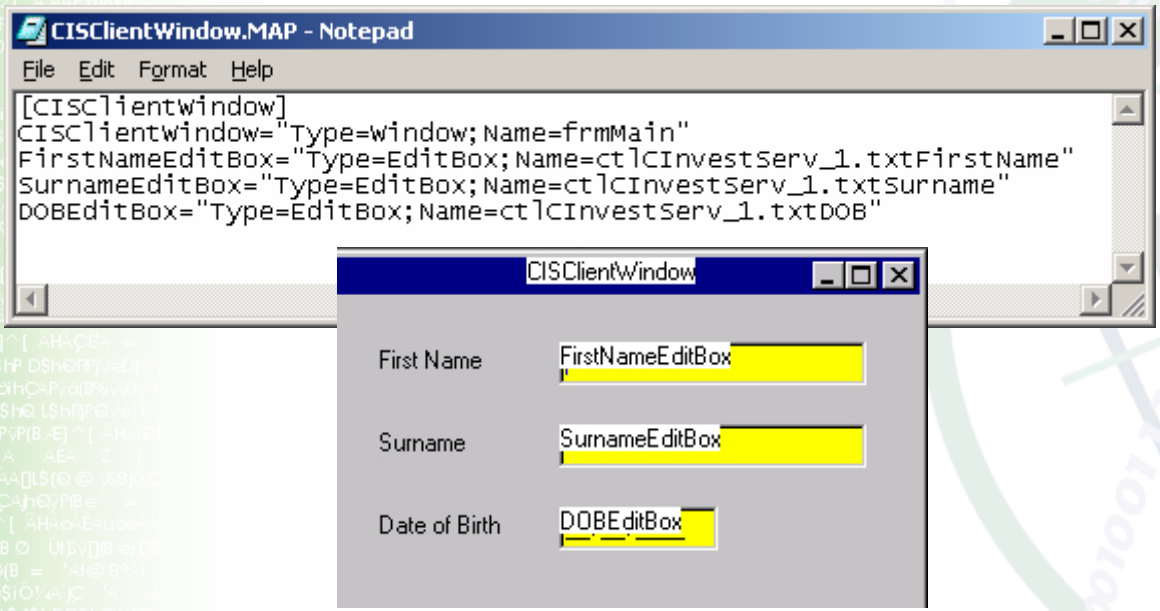
Action Words (RobotDDE)

- Carl Nagle has an Action Word framework for Rational Robot
  - http://members.aol.com/sascanagl/Default.htm
- Similar with a few minor variations
  - 2 levels to test clusters
  - Use of Application Map to map GUI elements to proper Ids
  - Predefined function on application map components



App Maps

- Map files are used to translate meaningful names for the objects and controls on the screen into technical names used by the automation tool to recognise the object



Low Level Test Tables (Steps)

- Step tables are required for each action
- All components on the user interface are linked to data and actions

2	C	SetApplicationMap	CISClientWindow.MAP		
3	C	Expressions	OFF		
4	**;RT**	**Window**	**Component**	**Action**	**Args_**
5					
6					
7	T	CISClientWindow	FirstNameEditBox	SetTextValue	^FirstNameEditBox
8	T	CISClientWindow	SurnameEditBox	SetTextValue	^SurnameEditBox
9	T	CISClientWindow	DOBEditBox	SetTextValue	^DOBEditBox
0					
1	C	CallStep	ResetCISClient		
2					
3					

Intermediate Test Tables (Suites)

- Setup the action calls

4					
5	**;RT**	**Steps**	**ARG**	**ARG**	**ARG**
6					
7					
8			^FirstNameEditBox = "RT-CIS01"	FirstNameEditBox	RT-CIS01
9			^SurnameEditBox = "TestCase"	SurnameEditBox	TestCase
10			^DOBEditBox = "19/04/1962"	DOBEditBox	19/04/1962
11	T	InputCISClient			
12			^navigateButton = "Forward"	navigateButton	Forward
13	T	QuotationNavigate			
14			^DateEditBox = "16/04/2003"	DateEditBox	16/04/2003
15			^LengthEditBox = "20"	LengthEditBox	20
16			^TargetRadioButton = "TRUE"	TargetRadioButton	TRUE
17			^PartialCheckBox = "Check"	PartialCheckBox	Check
18			^RegularWithdrawalCheckBox = "Check"	RegularWithdrawalC	Check
19	T	InputCISIllustration			
20			^navigateButton = "Forward"	navigateButton	Forward
21	T	QuotationNavigate			

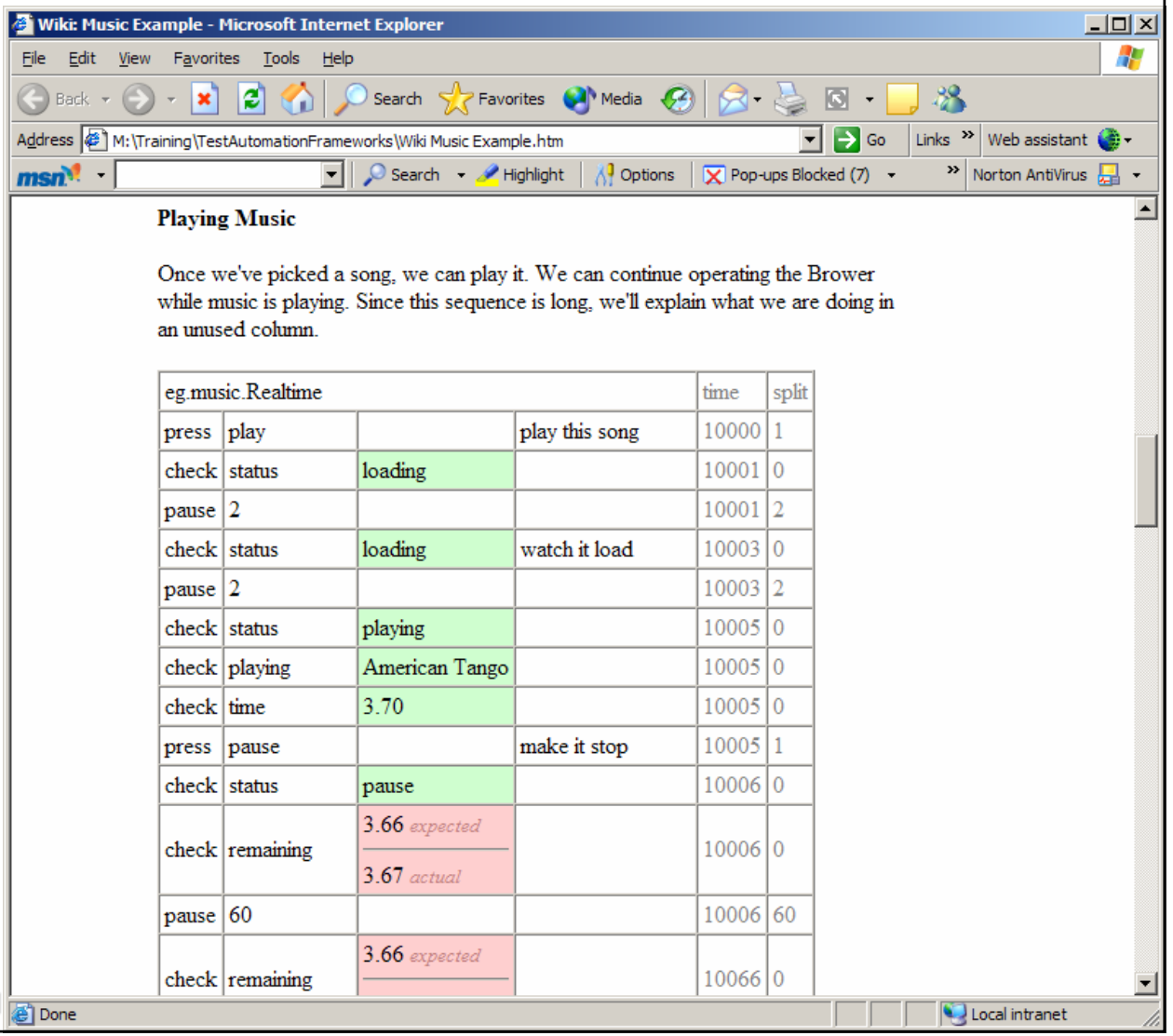
High-Level Test Tables (Cycles)

- Indicate suites to call

1	C	Version	1.0
2			
3	**;RT**	**SUITES**	**ARG**
4	T	RT-IGB01	
5	T	RT-IGB02	
6	T	RT-IGB03	
7	T	RT-IGB04	
8	T	RT-IGB05	
9	T	RT-IGB06	
10			

Framework for Integrated Test (FIT)
http://fit.c2.com/

- Test defined in HTML tables
- Designed to work with Wiki
- Execution extracts tests from tables
- Generated tables show test results
- Code distribution provided



Playing Music

Once we've picked a song, we can play it. We can continue operating the Brower while music is playing. Since this sequence is long, we'll explain what we are doing in an unused column.

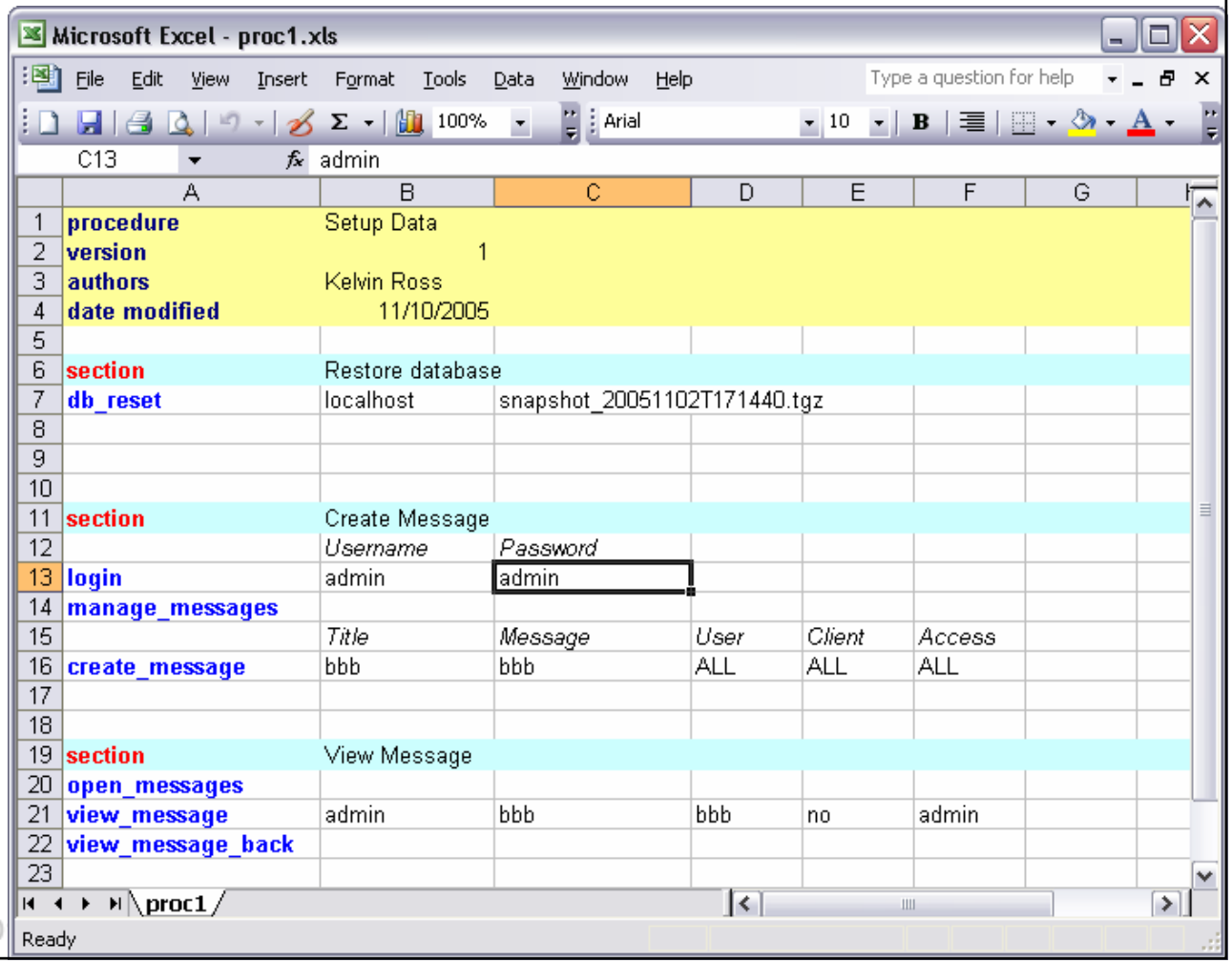
eg.music.Realtime				time	split
press	play		play this song	10000	1
check	status	loading		10001	0
pause	2			10001	2
check	status	loading	watch it load	10003	0
pause	2			10003	2
check	status	playing		10005	0
check	playing	American Tango		10005	0
check	time	3.70		10005	0
press	pause		make it stop	10005	1
check	status	pause		10006	0
check	remaining	3.66 *expected* 3.67 *actual*		10006	0
pause	60			10006	60
check	remaining	3.66 *expected*		10066	0

Overview

- Action Words
- Action Implementation
- SMARTLogger
- RDoc
- Exercise

Action Words

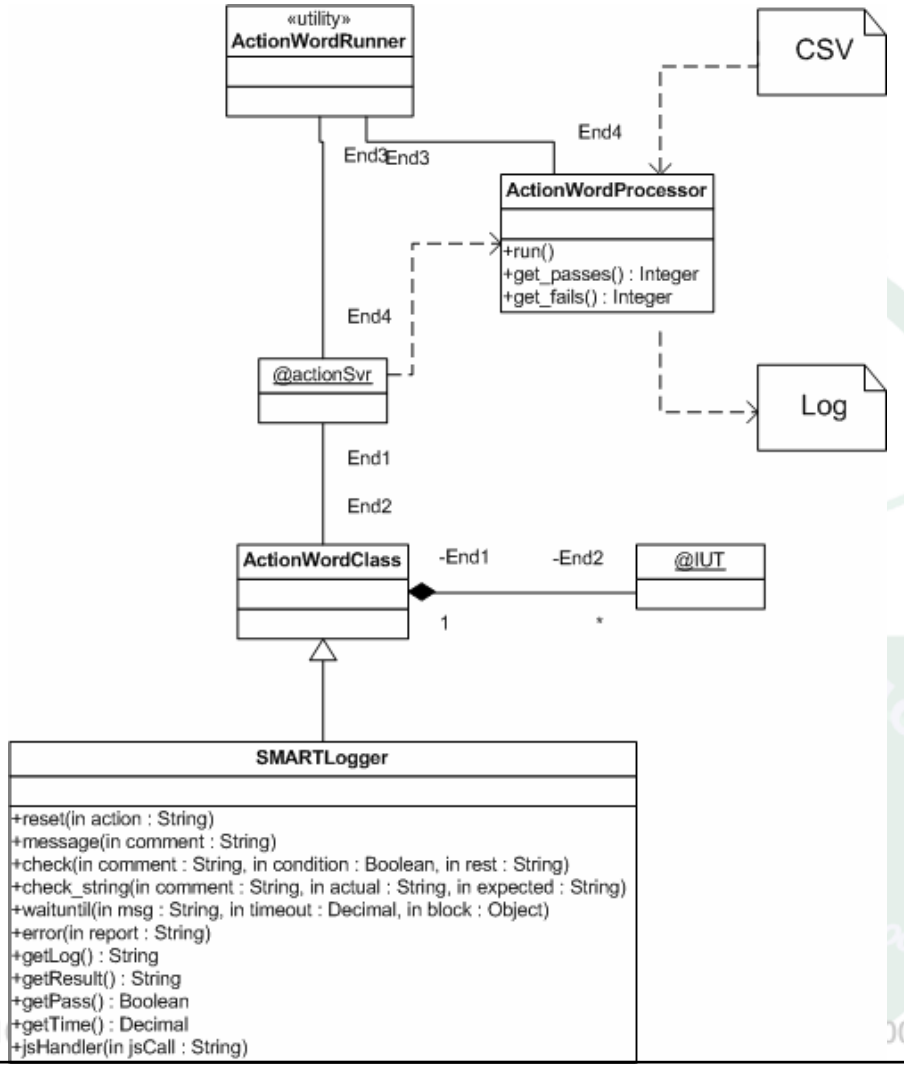
- Tester scripts actions with data
- Routines coded to implement test action with supplied data



	A	B	C	D	E	F
1	**procedure**	Setup Data				
2	**version**	1				
3	**authors**	Kelvin Ross				
4	**date modified**	11/10/2005				
5						
6	**section**	Restore database				
7	**db_reset**	localhost	snapshot_20051102T171440.tgz			
8						
9						
10						
11	**section**	Create Message				
12		*Username*	*Password*			
13	**login**	admin	admin			
14	**manage_messages**					
15		*Title*	*Message*	*User*	*Client*	*Access*
16	**create_message**	bbb	bbb	ALL	ALL	ALL
17						
18						
19	**section**	View Message				
20	**open_messages**					
21	**view_message**	admin	bbb	bbb	no	admin
22	**view_message_back**					
23						

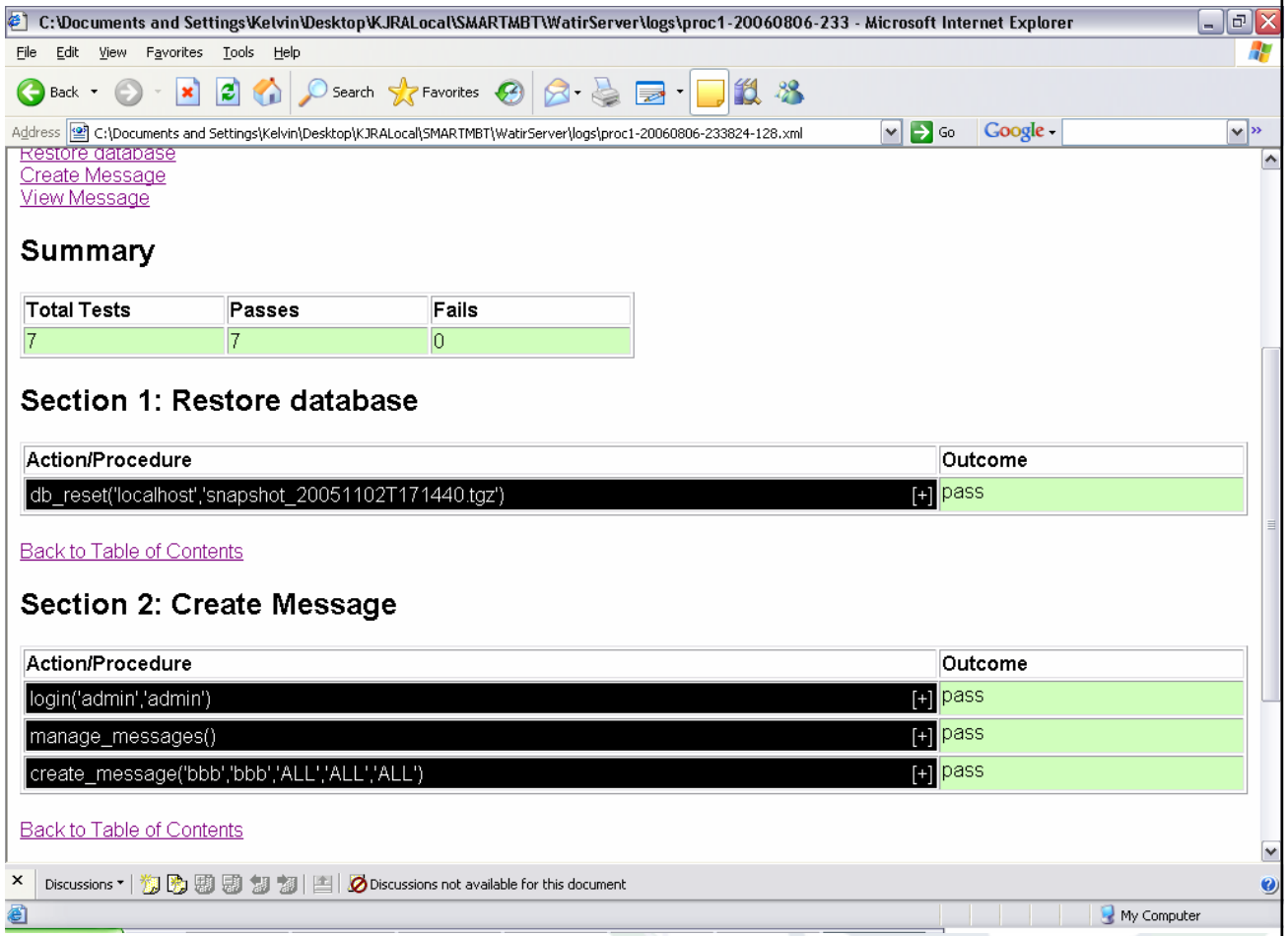
Action Words Architecture

- Action word libraries are created as classes
- Class methods are the action
- ActionWordRunner handles primitive action word constructs
  - E.g. section, …



Logging Test Results

- Output timestamped test logs as XML
- XSL translates into HTML navigable view



Overview

- Action Words
- **Action Implementation**
- SMARTLogger
- RDoc
- Exercise

About Ruby

- Simple: easy to learn and maintain
- Powerful
- Genuine object-oriented language
  - Everything you manipulate is an object
  - The results of those manipulations are themselves objects
- Rich libraries
- Rapid development
- Helpful community
- Open Source

- Useful links:
  - http://wtr.rubyforge.org/s101/doc/Ruby-cheat-sheet.doc
  - http://www.ruby-doc.org/
  - http://ruby.brian-schroeder.de/course/slides.pdf

Action Word Class

```
require 'SMARTLogger'
require 'CruiseControlWorking1'

class CCActions < SMARTLogger

attr_accessor :cc_impl

   def initialize
      super("")
      @cc_impl = CruiseControl.new
   end

private       # subsequent methods are private

public        # subsequent methods are public

   def start()
      check("Mode is off", @cc_impl.get_mode() == "off")
      @cc_impl.start()
      check("Final mode is inactive", @cc_impl.get_mode() == "inactive")
   end

end
@actionSvr = CCActions.new
```

Include SMARTLogger library

Include IUT library

Action word class inherits SMARTLogger

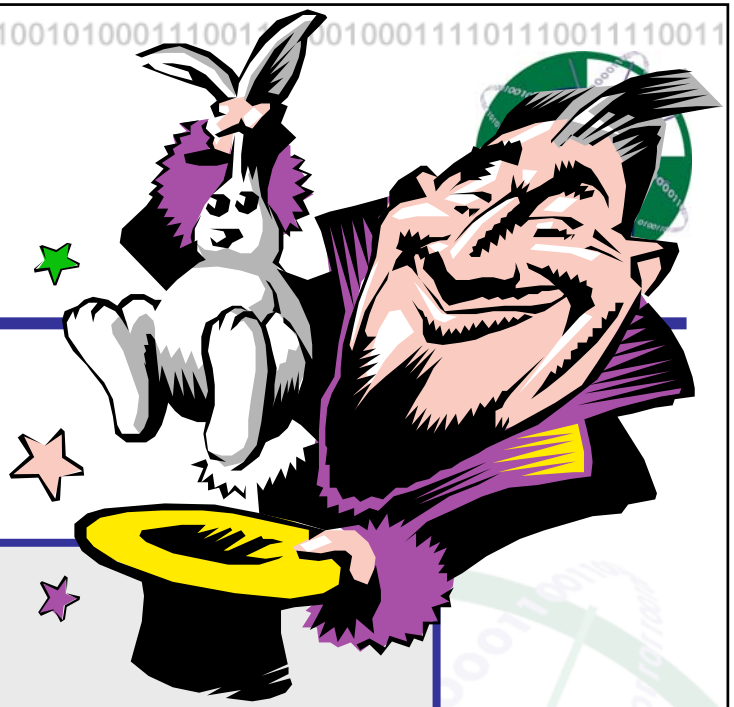
State is a handle on the IUT

Initialise creates a new IUT

Action words are methods. The action word does various checks interleaved with stimulating the IUT.

Server needs a handle on the class designated as @actionSvr

Demo



Cruise Control
Action Words Walkthrough

WATIR

- WATIR = Web Application Testing In Ruby
  - Petticord and Rogers
- Sits on top of Ruby
- Interfaces to Internet Explorer
  - Coordinates actions through invoking the COM interface which has access to IE's DOM
- http://wtr.rubyforge.org/

Sample Watir Action

- Initialisation

```
$smartcat_address = '192.168.25.129'

class SMARTCat < SMARTLogger

attr_accessor :ie,:base_url

    def initialize
      super("")
      @ie = IE.new
      sleep 1.0
      @base_url = 'http://' + $smartcat_address + '/smartcat/'
      @ie.set_fast_speed()
    end
```

Sample Watir Action

In fact 2 separate actions here!

- Action

```
def login_form(username, password)
      # Verify at correct starting page
      check("At welcome or login retry page", (@ie.title == "SMART CAT - Welcome")
                  || (@ie.title == "SMART CAT - Login"))

      # Action - fill out fields and submit
      @ie.text_field(:name, 'username').set(username)
      @ie.text_field(:name, 'password').set(password)
      @ie.button(:name, 'login').click
end


def login(username, password)
      message("Login to SMARTCat")
      # Submit login credentials
      login_form(username, password)

      # Verify logged in
      check("User found", @ie.pageContainsText("User:"))
      check("At menu page", @ie.title == "SMART CAT - Menu")
end
```

Overview

- Action Words
- Action Implementation
- SMARTLogger
- RDoc
- Exercise

SMARTLogger Methods

- def message(comment)
  - Adds a message to the test log
- def check(comment, condition,*rest)
  - Performs a check, like assert in XUnit
  - A failed check will fail the action
- def check_string(comment, actual, expected)
  - Performs a string comparison check
- def waituntil(msg, timeout, &block)
  - Continuously reevaluates the condition given in &block, until it is true or timeout expires. If timeout expires then the check fails.

Overview

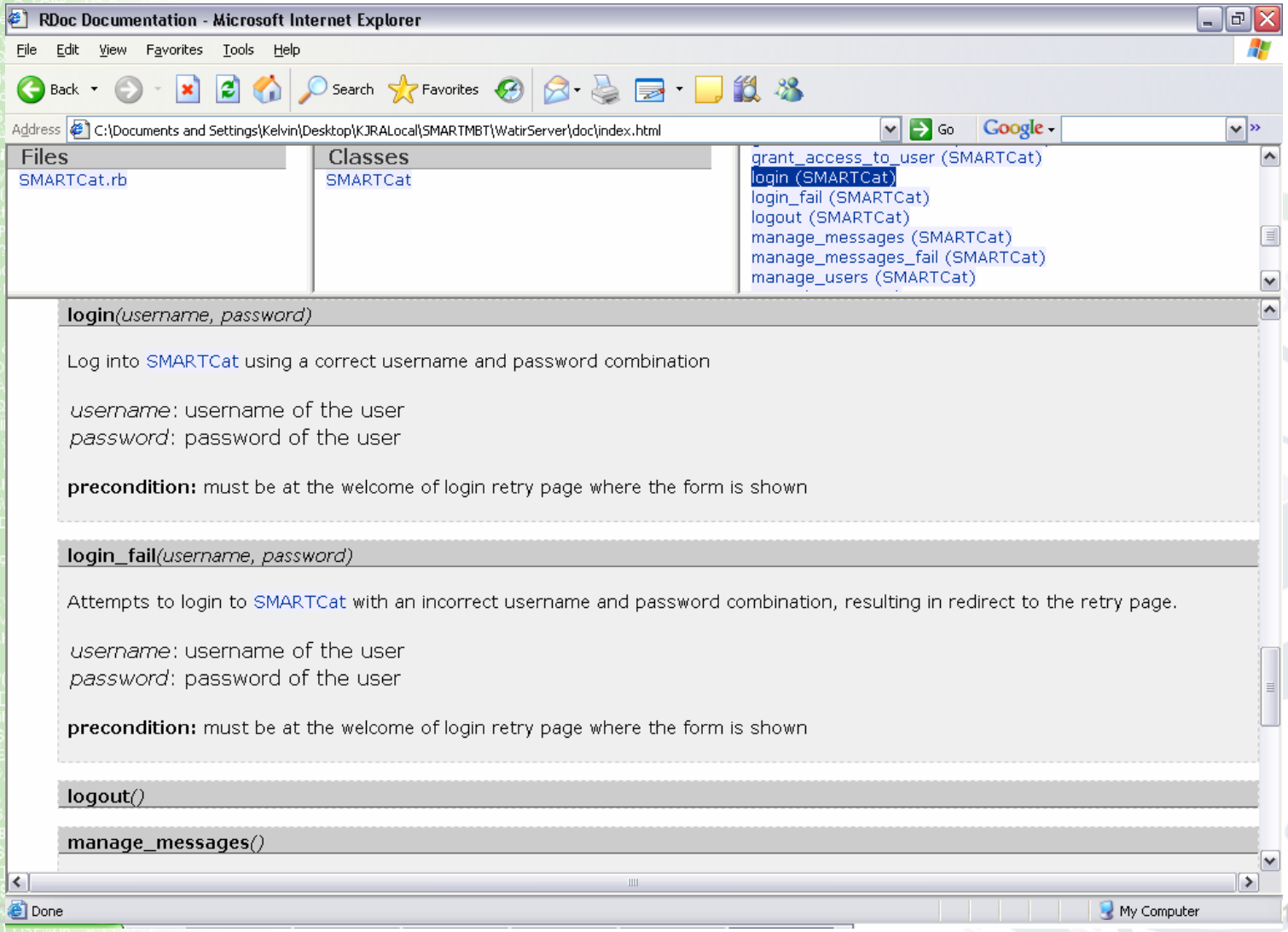
- Action Words
- Action Implementation
- SMARTLogger
- RDoc
- Exercise

RDoc

- Include comments prior to each method
- Run RDoc against the source file
- Generates a HTML representation

```
      # Log into SMARTCat using a correct username and password combination
      # _username_:: username of the user
      # _password_:: password of the user
      # <b>precondition:</b> must be at the welcome of login retry page where the form is shown
def login(username, password)
      message("Login to SMARTCat")
      # Submit login credentials
      login_form(username, password)

      # Verify logged in
      check("User found", @ie.pageContainsText("User:"))
      check("At menu page", @ie.title == "SMART CAT - Menu")
end
```

RDoc Output


RDoc Documentation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Kelvin\Desktop\KJRALocal\SMARTMBT\WatirServer\doc\index.html
Go Google
Files
SMARTCat.rb
Classes
SMARTCat
grant_access_to_user (SMARTCat)
login (SMARTCat)
login_fail (SMARTCat)
logout (SMARTCat)
manage_messages (SMARTCat)
manage_messages_fail (SMARTCat)
manage_users (SMARTCat)
login(username, password)
Log into SMARTCat using a correct username and password combination
username: username of the user
password: password of the user
precondition: must be at the welcome of login retry page where the form is shown
login_fail(username, password)
Attempts to login to SMARTCat with an incorrect username and password combination, resulting in redirect to the retry page.
username: username of the user
password: password of the user
precondition: must be at the welcome of login retry page where the form is shown
logout()
manage_messages()
Done
My Computer

Overview

- Action Words
- Action Implementation
- SMARTLogger
- RDoc
- Exercise

Exercise

SMART*Cat* Hands On